Novell exteNd Composer™

5.2.1

www.novell.com

CONNECT FOR J.D. EDWARDS* ONEWORLD* USER'S GUIDE

Novell®

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.

www.novell.com

Novell exteNd Composer 5.2.1 *Connect for J.D. Edwards OneWorld User's Guide*
March 2005

Online Documentation: To access the online documemntation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Indiana University Extreme! Lab Software License

JDOM.JAR

Phaos

W3C

Contents

About This Book

Purpose

This guide describes how to use the Novell exteNd Composer™ Connect for J.D. Edwards OneWorld. This product has design-time as well as runtime executables and uses J2EE Connector Architecture technology to provide integration capability.

Audience

This book is for developers and systems integrators who are planning to use Novell exteNd Composer to develop services and components for J.D. Edwards OneWorld.

Prerequisites

You should be familiar with the exteNd Composer work environment and deployment options. You also should be familiar with J.D. Edwards OneWorld. Familiarity with Java Connector Architecture is helpful but not required.

Software Versions

This guide assumes that you are using Novell exteNd Composer (Enterprise or Professional) version 5.2 (or higher) and that you are deploying your applications to a Java 2 Platform, Enterprise Edition (J2EE) Version 1.3 or equivalent application server.

The Novell exteNd Composer Connect for J.D. Edwards OneWorld supports J.D. Edwards OneWorld 7.3.2 and above.

Additional Documentation

For the complete set of Novell exteNd documentation, see the Novell Documentation Web site (http://www.novell.com/documentation-index/index.jsp).

1 Welcome to the Novell exteNd Composer Connect for J.D. Edwards OneWorld

Welcome to the *Novell exteNd Composer Connect for J.D. Edwards OneWorld User's Guide*. This Guide is a companion to the *Novell exteNd Composer User's Guide*, which describes how to use all the features of Composer except for the Connect Component Editors. You should be familiar with the *Composer User's Guide* before using this Guide.

Novell exteNd Composer™ provides separate Component Editors for each Connect, including the Connect for J.D. Edwards OneWorld. The special features of each component editor are described in separate Guides, like this one.

If you have been using exteNd Composer and are familiar with the core component editor (the XML Map Component Editor), then this Guide should be enough to get you started with the J.D. Edwards OneWorld Component Editor.

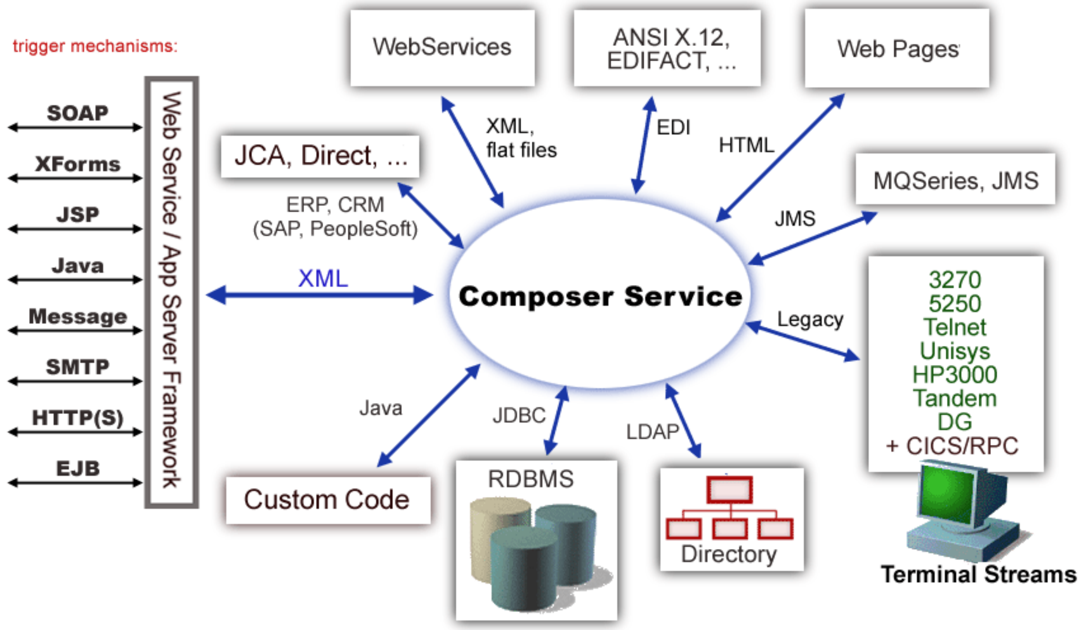
NOTE: To be successful with this Component Editor, you must be familiar with J.D. Edwards OneWorld and basic XML integration concepts.

About Novell exteNd Composer™

Novell exteNd Composer is the XML integration-broker component of the Novell exteNd suite. It encompasses a set of design tools for building XML integration applications and Web services, plus a runtime engine that enables execution and administration of the services that you build. The applications and services that you build with Composer can be deployed to any popular J2EE application server or servlet container. Supported application servers include JBoss, IBM WebSphere and BEA WebLogic in addition to the Novell exteNd application server. Apache Jakarta Tomcat is also supported. Check the Novell Web site for latest platform-support information.

At the core of Composer is a robust XML transformation engine capable of performing a wide range of data transformations, including joining of multiple documents, decomposition of documents, and creation of entirely new documents. The underlying enabling technologies include XSLT, XPath, ECMAScript, and Java. The Composer design environment offers a rich, intuitive graphical user interface, making it possible for you to specify XML transformations and mappings visually, using wizards, dialogs, and drag-and-drop gestures. You never have to write raw XSL or Java code.

Composer supports numerous kinds of data-source connectivity, through individual adapters called Connects. Using the functionality exposed in the various Connects, you can design EAI applications and Web services that pull data in from or push data out to different kinds of back-end systems, using a variety of transport protocols and technologies, ranging from 3270 and 5250 terminal data streams to Telnet, HP3000, Unisys T27 (and UTS), Tandem, and Data General, in addition to HTML screen-scraping, JMS messaging, and CICS RPC transactions. You can also take advantage of JDBC, LDAP, and other mechanisms to reach back-end data repositories and systems that might or might not natively understand XML. Composer Connects allow you to connect to these systems inobtrusively, so as to marshall non-XML data into XML form or vice versa without any need to modify host-system setups or code.



In addition to legacy data-stream and protocol-specific Connects, Composer has Connects for ERP and CRM systems, including Baan, PeopleSoft, SAP, Lawson, Oracle E-Business Suite, and Siebel. As with other Connect solutions, the ERP and CRM Connects are fully integrated into Composer's design-time environment so that you can use intuitive visual tools to create powerful custom integration solutions, eliminating the need to write Java code or edit raw XML or schemas by hand. You can also test the components that you build against live J.D. Edwards OneWorld connections, using the animation facility (step-through debugger) of the design environment. As part of the design and debug process, your Oracle-aware components can call other Composer Components (such as XML Map Components, JDBC Components, etc.) and make use of any of the core actions that Composer defines (such as Map, Function, Log, and other actions).

About the Composer Connect for J.D. Edwards OneWorld

The Novell exteNd Composer Connect for J.D. Edwards OneWorld allows you to build powerful XML-based integration solutions and enables you to reuse your existing J.D. Edwards OneWorld business functions with other applications—the key to building a successful e-business or integrated enterprise. The Composer Connect enables you to incorporate J.D. Edwards OneWorld business objects and services into new application initiatives.

You do not need to manually generate or install J.D. Edwards OneWorld XML schemas in order to use the Composer Connect for J.D. Edwards OneWorld. The Connect will generate schemas for you, as needed, automatically.

You also don't have to take any steps to preinstall RARs (resource adapter archives, defined by the Java Connector Architecture) on the target application server ahead of time. Composer handles RAR deployment for you automatically when you deploy any Composer-built service that utilizes the Composer Connect for J.D. Edwards OneWorld.

NOTE: Some one-time setup and configuration steps *are* required in order to use the Composer Connect for J.D. Edwards OneWorld. These steps are described in Chapter 2, "Getting Started With the Composer Connect for J.D. Edwards OneWorld".

About J2EE Connector Architecture

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting elements of the J2EE platform to a heterogeneous Enterprise Information System (EIS). Examples of EIS components include Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM), Supply Chain Management (SCM), mainframe transaction processing, database systems, and legacy applications that are not written in the Java programming language. By defining a common set of scalable, secure, transactional mechanisms reachable via a standard set of APIs, J2EE Connector Architecture enables the integration of an EIS with an application server and enterprise applications.

The J2EE Connector Architecture permits an EIS vendor to provide a standard resource adapter for its EIS. The resource adapter plugs into an application server, providing connectivity to an EIS, and integrating it with the rest of the enterprise. If an application server vendor has extended its system to support J2EE Connector Architecture, it is assured of seamless connectivity to multiple Enterprise Information Systems.

Before J2EE Connector Architecture, most EIS vendors offered vendor-specific architectures to provide connectivity between applications and their software. Each program interacting with an EIS needed to be hand-tooled by someone with a detailed knowledge of the target EIS. Custom software to provide connectivity across multiple systems was time consuming to develop, debug, and maintain.

By providing a standard set of APIs and contracts for managing connectivity, exposing EIS APIs, and using application-server services (like transaction control and connection pooling), J2EE Connector Architecture greatly reduces the need for custom programming. Developers can focus on business logic rather than connectivity and transaction-related logic and a variety of “plumbing issues.”

How J2EE Connector Architecture Works

The “major participants” in J2EE Connector Architecture include these components:

- Application server
- Resource adapter (RAR)
- Application

The application server is not strictly required. Certain services like connection pooling and transaction control will not be available in a “server” that is just a servlet container. But J2EE Connector Architecture resource adapters can still operate.

The RAR represents the interests of the underlying EIS.

The *application* interacts with the *resource adapter* using what J2EE Connector Architecture calls standard contracts. Standard contracts define what interactions are to take place and how they are exposed. The contract between the application and the resource adapter is called the Common Client Interface (CCI). The resource adapter, in turn, interacts with the application server under the Service Provider Interface (SPI), which defines how the management of resource adapter interactions occurs. The aspects of this include:

- Connectivity management
- Transaction demarcation
- Event listening (listeners can receive notification of significant events; for example, a connection failure)
- Pooling of connections and other resources

In the normal course of events, the application uses a *naming service* to locate the appropriate resource adapter. The application server supplies the naming service, and so it recognizes that a request is being made to locate a resource adapter. In such a case, the application server interposes a resource-adapter-supplied intermediate object that interacts between the resource adapter and the application server. Through this intermediating object, the application server manages the items within the SPI contract below the awareness of the application.

For more information about J2EE Connector Architecture, visit http://java.sun.com/j2ee/connector/.

About iWay Technology

Novell exteNd Composer uses licensed J2EE Connector Architecture adapter technology from iWay Software* (a division of Information Builders, Inc.) to mediate EIS interactions in the Composer Connect for J.D. Edwards OneWorld. A leader in the J2EE Connector Architecture technology space, iWay Software provides resource adapters and connectivity solutions across a wide array of EIS and other systems.

For more information about iWay, see http://www.iwaysoftware.com.

What Kinds of Applications Can You Build Using the Composer Connect for J.D. Edwards OneWorld?

With Composer Connect for J.D. Edwards OneWorld, you can build any kind of Web service or integration application that needs to push data into or pull data from a J.D. Edwards OneWorld-based data store using XML as the interchange format. Your integration application can be deployed to a J2EE application server and run as a public web service, or it can be used in "behind the firewall" scenarios. It can be triggered by a servlet, JSP, EJB, e-mail, timer, file arrival, JMS message arrival, or any of the supported Composer trigger types. It can also run standalone or as part of a workflow built using Composer Enterprise Edition's Process Manager. (For more information about deployment options, see "Deploying Your Project," in the Novell exteNd Composer User's Guide.)

2 Getting Started With the Composer Connect for J.D. Edwards OneWorld

This chapter describes how to set up and configure the exteNd Composer™ Connect for J.D. Edwards OneWorld.

Setup and Configuration

The following requirements must be completed before you can use the Composer Connect for J.D. Edwards OneWorld:

1. Determine the release number of your J.D. Edwards OneWorld system. The Novell exteNd Composer Connect for J.D. Edwards OneWorld supports J.D. Edwards OneWorld 7.3.2 and above.
2. Locate or acquire the J.D. Edwards JAR files. See "J.D. Edwards JAR Files" on page 15.
3. Update the classpath in **xconfig.xml,** in both the design environment and on the server, to reflect the addition of the J.D. Edwards JAR files. See "Adding the J.D. Edwards JAR Files to the Design-Time Environment" on page 16 and "Adding the J.D. Edwards JAR files to the Novell Application Server Environment" on page 16.
4. Update the license for your Composer Connect for J.D. Edwards OneWorld installation. See "Updating the Design-time Software License" on page 16 and "Updating the Runtime License" on page 17.
5. Create GenJava wrappers for J.D. Edwards business functions. See "Creating GenJava Wrappers" on page 18.

The steps necessary to accomplish these requirements are discussed in detail in the following sections.

J.D. Edwards JAR Files

Before attempting to use the Composer Connect for J.D. Edwards OneWorld (which is installed automatically as part of the default exteNd suite installation process), you must complete the installation by obtaining and installing two J.D. Edwards JAR files. *These files are proprietary to J.D. Edwards and are not shipped by Novell nor installed as part of the Novell exteNd Suite.*

You need the following jar files:

- Kernel.jar
- Connector.jar

These JAR files should be included in the \systems\classes directory in your J.D. Edwards environment. If they are not, contact your J.D. Edwards representative to obtain copies of the JAR files.

When you have located the J.D. Edwards JAR files, you must add the JAR files to the classpath in both your design-time and runtime environments, as described in the following sections.

Adding the J.D. Edwards JAR Files to the Design-Time Environment

➢ To add the J.D. Edwards JAR Files to your design-time configuration

1. If Composer is running, shut it down before proceeding.
2. Obtain the J.D. Edwards JAR files (see "J.D. Edwards JAR Files" on page 15), if it is not included in your J.D. Edwards OneWorld installation.
3. Copy the J.D. Edwards JAR files to the **/Common/lib** folder of the exteNd installation directory on your design-time computer.

➢ To add the J.D. Edwards JAR Files to the design-time classpath

1. Locate your **xconfig.xml** file under **/Composer/Designer/bin** and open the file in a text editor.
2. Scroll to the bottom. Within the <RUNTIME> element, you should see many <JAR> entries.
3. Add additional <JAR> elements that specify the path to the J.D. Edwards JAR files, as follows:

   ```
   <JAR>..\..\..\Common\lib\Kernel.jar</JAR>
   <JAR>..\..\..\Common\lib\Connector.jar</JAR>
   ```

   These entries tell the class loader where it can find the J.D. Edwards JAR files.
4. Save and close **xconfig.xml**.

Adding the J.D. Edwards JAR files to the Novell Application Server Environment

1. If the application server is running, shut it down before proceeding.
2. Obtain the J.D. Edwards JAR files (see "J.D. Edwards JAR Files" on page 15), if they are not included in your J.D. Edwards OneWorld installation.
3. Copy the J.D. Edwards JAR files to a suitable location on your application server. The exact location doesn't matter, as long as you create a classpath entry pointing to the J.D. Edwards JAR files location, as described in the following steps.
4. Update the application server classpath. For the Novell exteNd application server, locate the **AgJars.conf** file in the **AppServer/bin** directory and open the file in a text editor.
5. Create new entries in the "MODULE COMMON" section, as follows:

   ```
   $SS_LIB ../../Common/lib/Kernel.jar
   $SS_LIB ../../Common/lib/Connector.jar
   ```

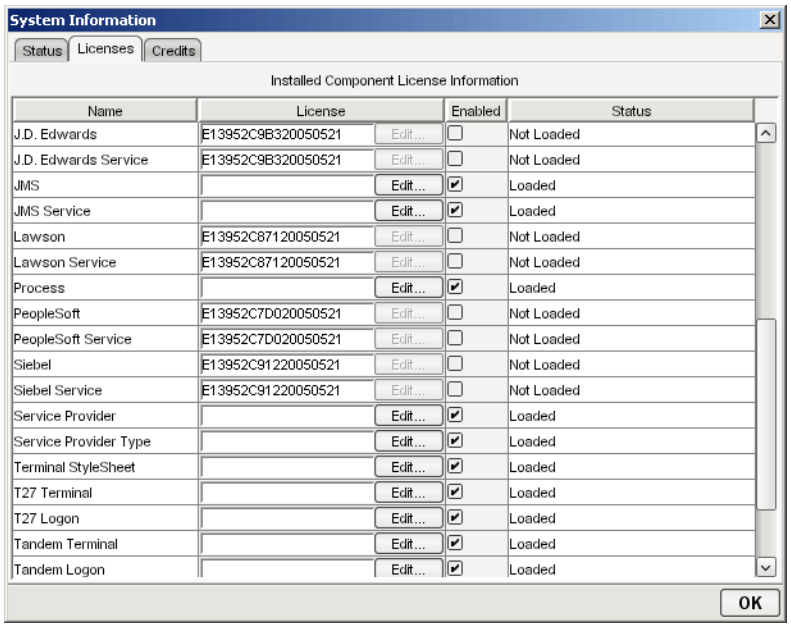
   This example assumes that you have placed the JAR files in the application server **/Common/lib** directory. Edit this path as required to reflect the actual target directory.

   NOTE: For application servers other than Novell exteNd, follow the application server vendor's instructions for updating the classpath.

Updating the Design-time Software License

1. Obtain a valid license string from your Novell representative.

   NOTE: You use the same license string to activate the design-time and runtime versions of Composer Connect for J.D. Edwards OneWorld.
2. Start Composer.
3. Select **About Composer** from the **Help** menu. The About Extend Composer dialog box is displayed.
4. Select **System**. The System Information dialog box is displayed.
5. Click on the **Licenses** tab, located in the upper left corner of the System Information dialog box. The license information for your Composer installation is displayed.



Installed Component License Information

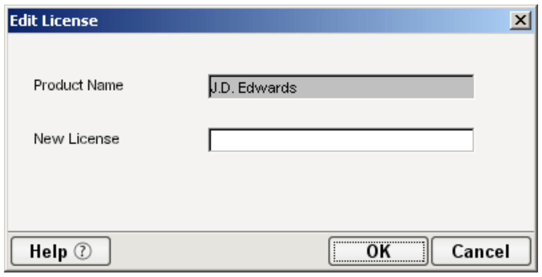
Name	License		Enabled	Status
J.D. Edwards	E13952C9B320050521	Edit...	☐	Not Loaded
J.D. Edwards Service	E13952C9B320050521	Edit...	☐	Not Loaded
JMS		Edit...	✓	Loaded
JMS Service		Edit...	✓	Loaded
Lawson	E13952C87120050521	Edit...	☐	Not Loaded
Lawson Service	E13952C87120050521	Edit...	☐	Not Loaded
Process		Edit...	✓	Loaded
PeopleSoft	E13952C7D020050521	Edit...	☐	Not Loaded
PeopleSoft Service	E13952C7D020050521	Edit...	☐	Not Loaded
Siebel	E13952C91220050521	Edit...	☐	Not Loaded
Siebel Service	E13952C91220050521	Edit...	☐	Not Loaded
Service Provider		Edit...	✓	Loaded
Service Provider Type		Edit...	✓	Loaded
Terminal StyleSheet		Edit...	✓	Loaded
T27 Terminal		Edit...	✓	Loaded
T27 Logon		Edit...	✓	Loaded
Tandem Terminal		Edit...	✓	Loaded
Tandem Logon		Edit...	✓	Loaded

6 Scroll down until you see the row for J.D. Edwards.

NOTE: The Connect ships with an Evaluation license string which may be used for 90 days.

7 To use the evaluation license string, select the check box in the **Enabled** column and skip to Step 10. To enter a different license string, select the check box in the **Enabled** column. The **Edit** button in the J.D. Edwards row is enabled.

8 Select **Edit** in the J.D. Edwards row. The **Edit License** dialog box is displayed.



Product Name	J.D. Edwards
New License	

9 Type the license string for the Composer Connect for J.D. Edwards OneWorld in the **New License** field.

10 Scroll down until you see the row for J.D. Edwards Service.

11 Repeat Step 7 through Step 9 for the J.D. Edwards Service row.

12 Exit out of all dialogs by selecting **OK**.

Updating the Runtime License

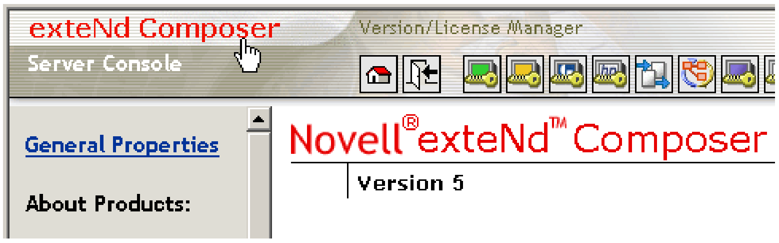
1 Make sure that the Novell exteNd Composer Enterprise Server is installed.

2 Start the application server. The Novell exteNd Composer Enterprise Server starts automatically when you start the application server.

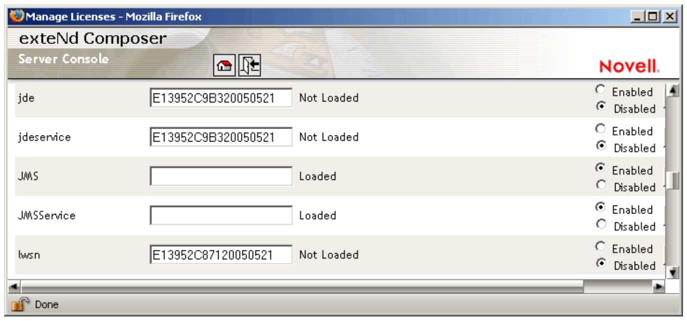
3 Open a browser window and navigate to the Novell exteNd Composer main administrative console. Typically, this is at:

```
http://localhost/exteNdComposer
```

4 In the upper left corner of the console window, click the **exteNd Composer** logo immediately above the words "Server Console," as shown in the following illustration.



5 In the content area of the main frame, near the bottom, click **Licenses**. The Manage Licenses window is displayed. This window shows the license status of every Composer Connect. See the following illustration.



jde	E13952C9B320050521	Not Loaded	Enabled [✓ Disabled]
jdeservice	E13952C9B320050521	Not Loaded	Enabled [✓ Disabled]
JMS		Loaded	[✓ Enabled] Disabled
JMSService		Loaded	[✓ Enabled] Disabled
lwsn	E13952C87120050521	Not Loaded	Enabled [✓ Disabled]

6 Scroll down to the entry labeled "jde".

NOTE: The Connect ships with Evaluation license strings which may be used for 90 days.

7 Type a license string in the text field in the "jde" row; then select **Enabled** in the "jde" row. To use the evaluation license string, just select **Enabled.**

8 Select **Update** in the "jde" row.

9 Scroll down to the entry labeled "jdeservice".

10 Type a license string in the text field in the "jdeservice" row; then select **Enabled** in the "jdeservice" row. To use the evaluation license string, just select **Enabled.**

11 Select **Update** in the "jdeservice" row.

Creating GenJava Wrappers

Before you can use the Composer Connect for J.D. Edwards OneWorld, you must create GenJava wrappers. J.D. Edwards OneWorld provides a utility, GenJava, that generates Java wrappers for the business functions running as part of the OneWorld server. Once GenJava is executed, it creates Java class files for all the interface classes and associated data structures. GenJava is supplied as a command line process with several run-time options. For more information about GenJava, see the J.D. Edwards Interoperability Guide for OneWorld Xe.

3 Creating a J.D. Edwards OneWorld Component

To create a Component that utilizes the exteNd Composer™ Connect for J.D. Edwards OneWorld, you need to do three things:

- Create a Connection Resource (to allow your component to connect to a J.D. Edwards OneWorld system)
- Create any necessary XML Templates
- Create the Component itself (containing your business logic)

Each of these processes is discussed in detail in this chapter.

Creating a Connection Resource

Before creating a component that interacts with a J.D. Edwards OneWorld system, you need to create a Connection Resource, which is a lightweight Composer object (xObject) that encapsulates basic connection information (parameter values) associated with a connection to a back-end system.

In addition to a connection resource, a J.D. Edwards OneWorld Component requires that you have already created XML templates so that you have sample input and output documents for use in designing your component. For more information, see "Creating an XML Template" in the *Novell exteNd Composer User's Guide.*

If your component design calls for any other resources, such as custom scripts, XSL, XSD, etc., you should create these before creating the J.D. Edwards OneWorld Component. For more information, see *"*Creating Custom Scripts" in the *Novell exteNd Composer User's Guide*.

Types of Connection Resources

You create different types of connection resource depending on the type of interaction with the J.D. Edwards OneWorld system that is desired. If your application needs to initiate J.D. Edwards OneWorld business events, you need to create a J.D. Edwards OneWorld Connection resource. If your application needs to process data when a business event occurs within the J.D. Edwards OneWorld system, you need to create an inbound connection resource. Inbound connection resources include file and TCP connection types.

About Constant-Driven and Expression-Driven Connection Parameters

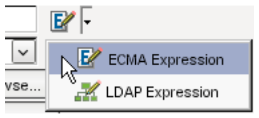
You can specify Connection parameter values in one of two ways: as Constants or as Expressions. A constant-driven parameter uses the value you type in the Connection dialog every time the Connection is used. An expression-driven parameter allows you to set the value using a programmatic expression, which can result in a different value each time the connection is used at runtime. This allows the Connection's behavior to be flexible and vary based on runtime conditions each time it is used.

For instance, one very simple use of an expression-driven parameter in a Connection would be to define the User ID and Password as PROJECT Variables (e.g. PROJECT.XPATH("USERCONFIG/MyDeployUser"). This way when you deploy the project, you can update the PROJECT Variables in the Deployment Wizard to values appropriate for the final deployment environment. At the other extreme, you could have a custom script that queries a Java business object in the Application Server to determine what User ID and Password to use.

➤ **To switch a parameter from Constant-driven to Expression driven:**

1 Click the right mouse button in the parameter field you are interested in changing.

2 Select **Expression** from the context menu and the editor button will appear or become enabled.



3 Click on the button and then create an expression that evaluates to a valid parameter value at runtime. (Strings should be wrapped in double-quotes.)

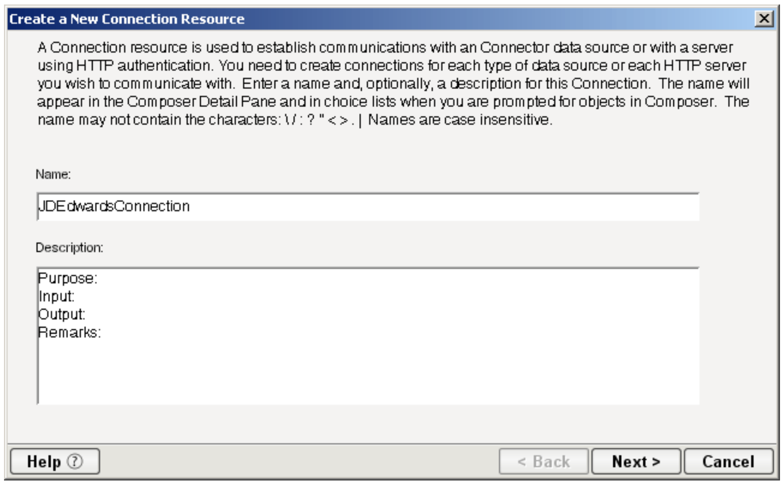
Creating a J.D. Edwards OneWorld Connection Resource

➤ **To Create a J.D. Edwards OneWorld Connection Resource:**

1 Select **File** > **New** > **xObject**. The New xObject dialog box is displayed.

2 Select the **Resource** tab.

3 Double-click on **Connection**. The "Create a New Connection Resource" wizard is displayed.



A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive.

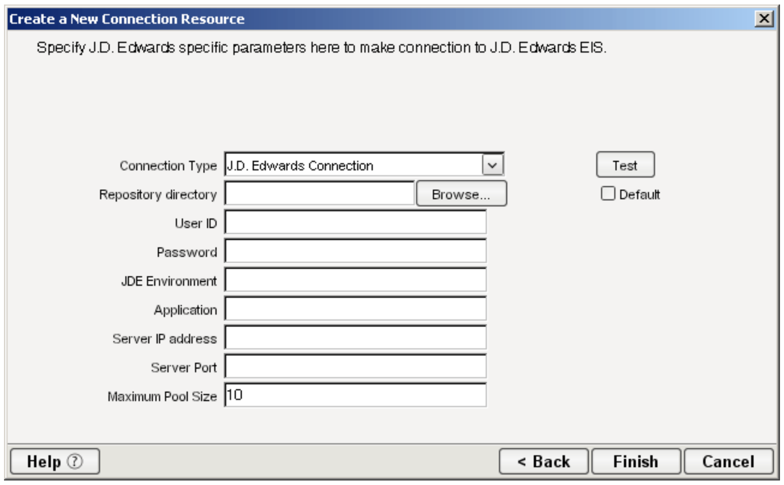
Name: JDEdwardsConnection

Description:
Purpose:
Input:
Output:
Remarks:

4 Type a **Name** for the connection object.

5 Optionally, type **Description** text.

6 Select **Next**. A connection parameters panel is displayed.



Specify J.D. Edwards specific parameters here to make connection to J.D. Edwards EIS.

Connection Type	J.D. Edwards Connection
Repository directory	
User ID	
Password	
JDE Environment	
Application	
Server IP address	
Server Port	
Maximum Pool Size	10

7 Select **J.D. Edwards Connection** from the **Connection Type** list.

8 Type the path to the J.D. Edwards repository directory in the **Repository directory** field, or select Browse to display a dialog box that you use to select the directory.

The repository directory is the directory in which the Java files created by the OneWorld GenJava program are located (see "Creating GenJava Wrappers" on page 18).

9 Type a J.D. Edwards OneWorld user ID in the **User ID** field.

10 In the **Password** field, type the password for the user specified in the User ID field.

11 Type the name of the J.D. Edwards OneWorld environment (e.g., DU7333) in the **JDE Environment** field. See your J.D. Edwards OneWorld documentation for more information about this parameter.

12 Type the name of the J.D. Edwards OneWorld application in the **Application** field.

13 Type the IP address of the J.D. Edwards OneWorld server in the **Server IP address** field.

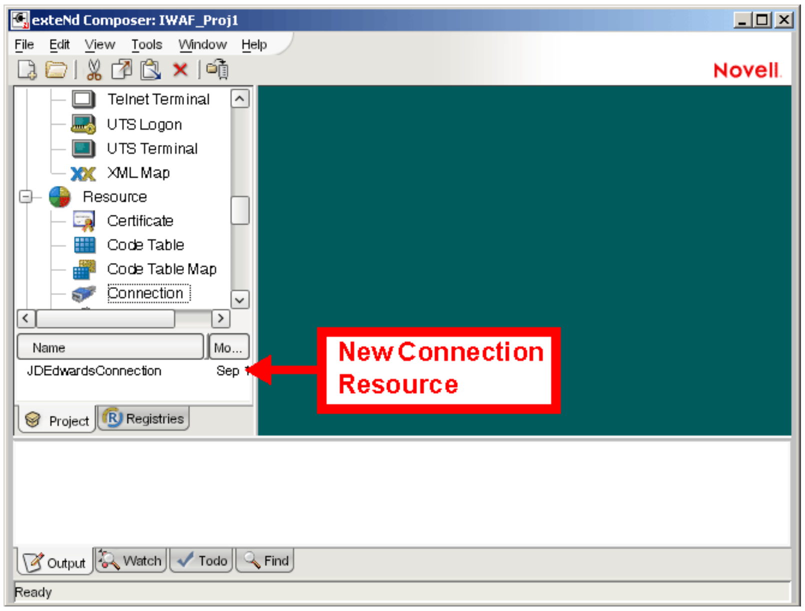
14 Type the number of the port on which the server is listening in the **Server Port** field. The default port is 6009.

15 Type the desired maximum pool size in the **Maximum Pool Size** field. This is a JCA setting that specifies the maximum number of connections to be allowed in the connection pool (not applicable for non-managed environments).

16 If desired, select **Test** to see if your connection parameters and network environment allow you to create a live connection. Composer displays a message indicating the success or failure of the test. The test connection is discarded immediately after the test. *You can continue working with the connection resource, even if the connection fails.*

NOTE: This test does not test the connection pool.

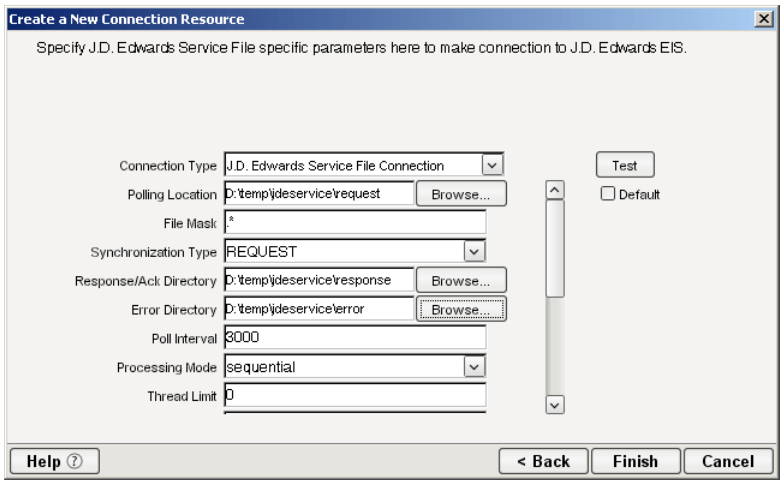
17 Select **OK**. The newly-created connection resource appears in the Composer Connection Resource detail pane.



Creating a File Service Connection Resource

➢ **To Create a J.D. Edwards Service File Connection Resource:**

1. Select **File > New > xObject**. The New xObject dialog box is displayed.
2. Select the **Resource** tab.
3. Double-click on **Connection**. The "Create a New Connection Resource" wizard is displayed.
4. Type a name for the connection object in the **Name** field.
5. Optionally, type a description of the connection object in the **Description** field.
6. Select **Next**. A connection parameters panel is displayed.


Create a New Connection Resource

Specify J.D. Edwards Service File specific parameters here to make connection to J.D. Edwards EIS.

Connection Type	J.D. Edwards Service File Connection
Polling Location	D:\temp\jdeservice\request
File Mask	.*
Synchronization Type	REQUEST
Response/Ack Directory	D:\temp\jdeservice\response
Error Directory	D:\temp\jdeservice\error
Poll Interval	3000
Processing Mode	sequential
Thread Limit	0

7. Select **J.D. Edwards Service File Connection** from the **Connection Type** list.
8. Use the **Browse** button next to the **Polling Location** field to select the target file system location for the Siebel XML file.
9. In the **File Mask** field type the file name to be used for the output file generated as a result of this operation. The default is ".*".

10 Select REQUEST, REQUEST_RESPONSE, or REQUEST_ACK from the **Synchronization Type** list. Select the option that is correct for the system with which you are integrating (e.g., if your system is set up to process a response, select REQUEST_RESPONSE). The Synchronization Type selected also determines the tabs that are displayed in the Service Request Native Environment pane (e.g., if REQUEST_RESPONSE is selected, both Request and Response tabs are displayed).

11 Use the **Browse** button next to the **Response/Ack Directory** field to select the directory to which responses or acknowledgements from your application will be written.

12 Use the **Browse** button next to the **Error Directory** field to select the directory to which error information will be written.

13 In the **Poll Interval** field, type the interval (in milliseconds) in which the Polling Location is checked for input. The default is 3000 (3 seconds).

14 Select Threaded or Sequential from the **Processing Mode** list. Threaded indicates processing of multiple requests simultaneously. Sequential indicates serial processing of requests.

15 If Threaded processing mode is selected in the Processing Mode list, use the **Thread Limit** field to specify the maximum number of requests that can be processed simultaneously.

16 From the **Metadata Connection** list, select a J.D. Edwards OneWorld connection resource from the list of defined connection resources. This connection is used for acquiring metadata from the J.D. Edwards OneWorld system.

17 Type the desired maximum pool size in the **Maximum Pool Size** field. This is a JCA setting that specifies the maximum number of connections to be allowed in the connection pool (not applicable for non-managed environments).

Creating an TCP Service Connection Resource

➢ **To Create a J.D.Edwares Service HTTP Connection Resource:**

1 Select **File** > **New** > **xObject**. The New xObject dialog box is displayed.

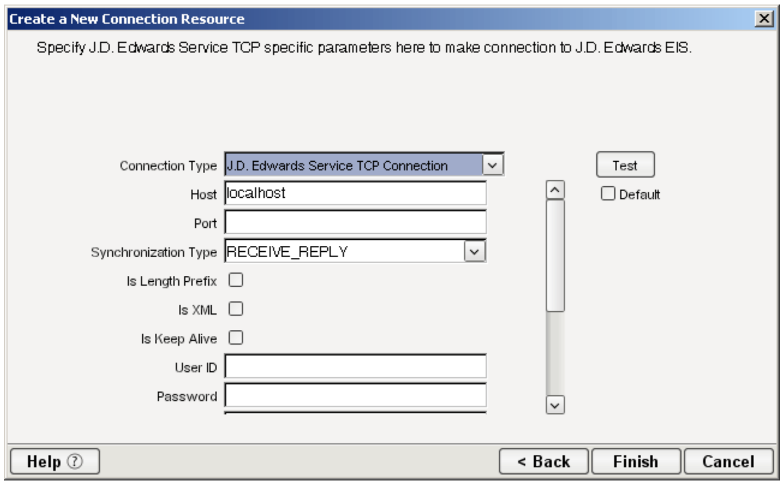
2 Select the **Resource** tab.

3 Double-click on **Connection**. The “Create a New Connection Resource” wizard is displayed.

4 Type a name for the connection object in the **Name** field.

5 Optionally, type a description of the connection object in the **Description** field.

6 Select **Next**. A connection parameters panel is displayed.


Create a New Connection Resource

Specify J.D. Edwards Service TCP specific parameters here to make connection to J.D. Edwards EIS.

Connection Type	J.D. Edwards Service TCP Connection
Host	localhost
Port	
Synchronization Type	RECEIVE_REPLY
Is Length Prefix	☐
Is XML	☐
Is Keep Alive	☐
User ID	
Password	

7 Select **J.D. Edwards Service TCP Connection** from the **Connection Type** list.

8 In the **Host** field, specify the host name or URL of the system on which the J.D. Edwards OneWorld system is installed.

9 In the **Port** field, type the name of the port on which the host is listening.

10 Select RECEIVE_REPLY, RECEIVE_ACK, or RECEIVE from the **Synchronization Type** list:.

- Select RECEIVE_REPLY if the J.D. Edwards OneWorld application expects a reply sent back to it.
- Select RECEIVE_ACK when a TCP/IP acknowledgement (ACK) is sent back to the J.D. Edwards OneWorld application.
- Select RECEIVE if the J.D. Edwards OneWorld application does not expect a return.

11 Select **Is Length Prefix** if the J.D. Edwards OneWorld application sends data that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port. When this option is selected, deselect the **Is XML** option.

12 Select **Is XML** if the J.D. Edwards OneWorld application sends data in XML format. When this option is selected, deselect the **Is Length Prefix** checkbox.

13 Select **Is Keep Alive** to maintain continuous communication between the J.D. Edwards OneWorld application and your application.

14 From the **Metadata Connection** list, select an outbound J.D. Edwards OneWorld connection resource from the list of defined connection resources. This connection is used for acquiring metadata from the J.D. Edwards OneWorld system.

15 Type the desired maximum pool size in the **Maximum Pool Size** field. This is a JCA setting that specifies the maximum number of connections to be allowed in the connection pool (not applicable for non-managed environments).

XML Templates for J.D. Edwards OneWorld Components

Before creating a J.D. Edwards OneWorld Component, you should create any XML templates that might be needed. XML templates are used to tell Composer which XML sample documents (e.g., input, output, scratch pad, fault, XSD, or DTD) to use. Once you've specified the XML templates, you can create a component that uses the sample documents to represent the inputs and outputs processed by your component. For more information about XML templates, see "Creating a New XML Template" in the *Novell exteNd Composer User's Guide.*

Creating J.D. Edwards OneWorld Components

As part of the process of creating a J.D. Edwards OneWorld Component, you can select an existing J.D. Edwards OneWorld connection resource, or you can create a new one. If you create the connection beforehand, it is available for use by any J.D. Edwards OneWorld Components in the current project. *If you have not already created at least one J.D. Edwards OneWorld connection resource in the current project, you will be prompted to do so when you try to create a J.D. Edwards OneWorld Componen*t.

NOTE: You can still create a Component without first creating a Connection Resource, but you won't be able to use any debug features that depend on a live connection.

➢ **To Create a New J.D. Edwards OneWorld Component:**

1 Select **File>New>xObject**. The New xObject dialog box is displayed.

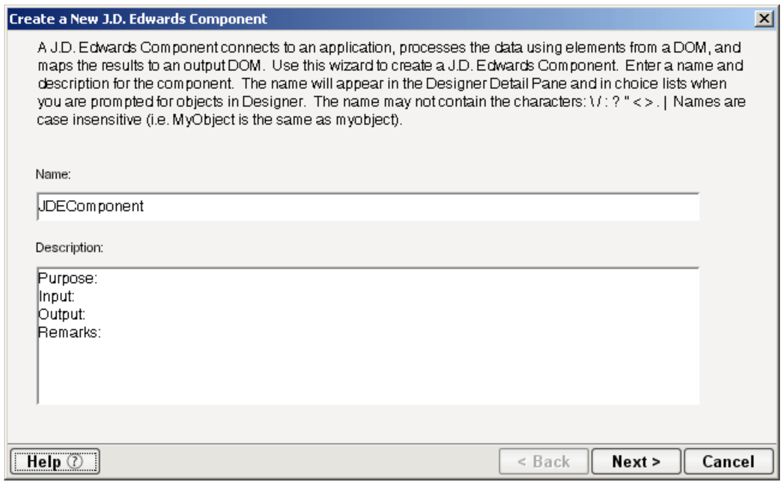
NOTE: Alternatively, under **Component** in the Composer Navigator pane (Explorer view), you can highlight **J.D. Edwards**, click the right mouse button, then select **New**.

2 Select the **Component** tab.

3 Double-click on "J.D. Edwards."

If you have not previously defined a connection resource, you are prompted to do so now (see "Creating a Connection Resource" on page 19).

If you have already defined a connection resource, the "Create a New J.D. Edwards Component" Wizard is displayed.



A J.D. Edwards Component connects to an application, processes the data using elements from a DOM, and maps the results to an output DOM. Use this wizard to create a J.D. Edwards Component. Enter a name and description for the component. The name will appear in the Designer Detail Pane and in choice lists when you are prompted for objects in Designer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive (i.e. MyObject is the same as myobject).

Name:

JDEComponent

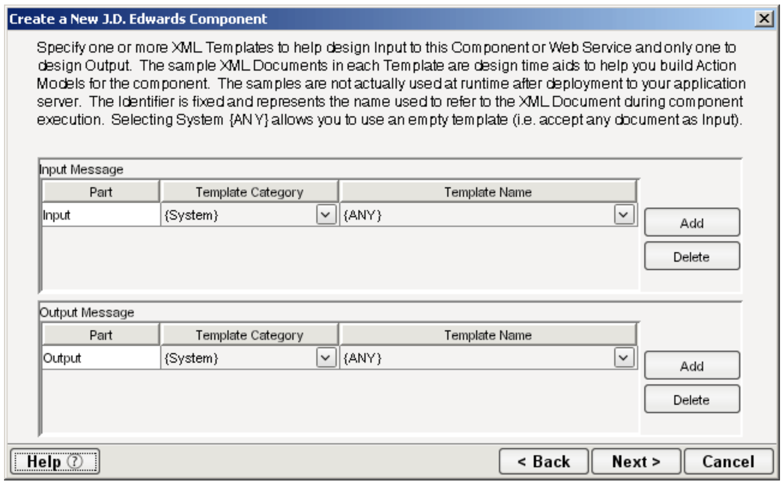
Description:

Purpose:
Input:
Output:
Remarks:

4 Type a **Name** for the new J.D. Edwards Component.

5 Optionally, type **Description** text.

6 Select **Next**. The XML Input/Output Message Property Info panel of the New J.D. Edwards Component Wizard is displayed.



Specify one or more XML Templates to help design Input to this Component or Web Service and only one to design Output. The sample XML Documents in each Template are design time aids to help you build Action Models for the component. The samples are not actually used at runtime after deployment to your application server. The Identifier is fixed and represents the name used to refer to the XML Document during component execution. Selecting System {ANY} allows you to use an empty template (i.e. accept any document as Input).

Input Message

Part	Template Category	Template Name
Input	{System}	{ANY}

Output Message

Part	Template Category	Template Name
Output	{System}	{ANY}

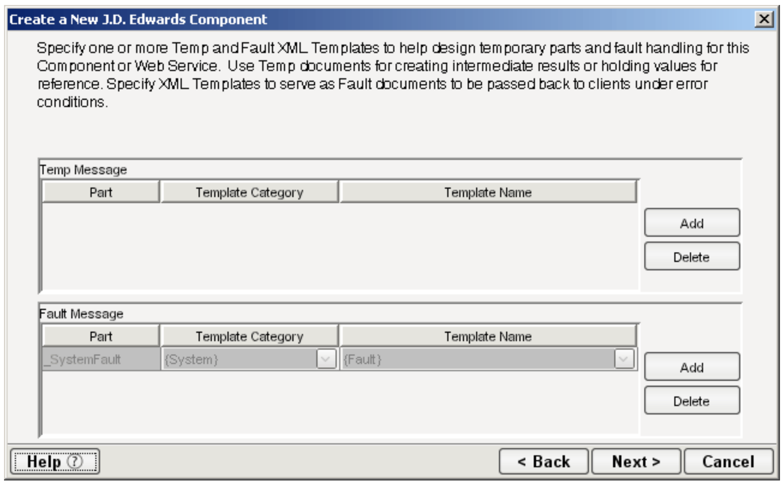
7 Specify the Input and Output templates as follows:

- Type a name for the template under **Part** if you wish the name to appear in the DOM as something other than "Input"
- Select a **Template Category** if it is different than the default category.
- Select a **Template Name** from the list of XML templates in the selected **Template Category**.
- To add additional input XML templates, click **Add**, then choose a **Template Category** and **Template Name** for each.
- To remove an input XML template, select an entry and click **Delete**.

8 Select an XML template for use as an Output DOM using the procedure described in the previous step.

NOTE: You can specify an input or output XML template that contains no structure by selecting {System}{ANY} as the Input or Output template. For more information, see "Creating an Output Document without Using a Template" in the *Novell exteNd Composer User's Guide*.

9 Select **Next**. The Temp and Fault XML Template panel is displayed.


Create a New J.D. Edwards Component

Specify one or more Temp and Fault XML Templates to help design temporary parts and fault handling for this Component or Web Service. Use Temp documents for creating intermediate results or holding values for reference. Specify XML Templates to serve as Fault documents to be passed back to clients under error conditions.

Temp Message

Part	Template Category	Template Name

Fault Message

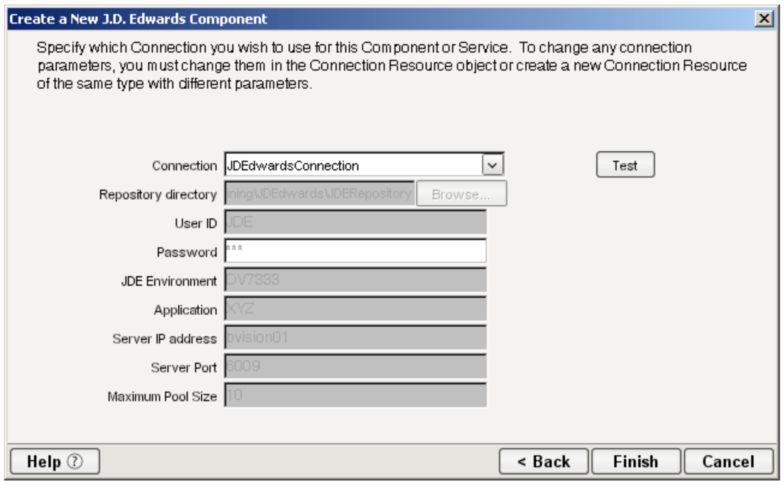
Part	Template Category	Template Name
_SystemFault	{System}	{Fault}

10 If desired, specify a template to be used as a scratch pad under the "Temp Message" pane of the dialog box. This can be useful if you need a place to hold values that will be used temporarily during the execution of your component. Select a **Template Category** if it is different than the default category. Then select a **Template Name** from the list of XML templates in the selected Template Category.

11 Under the "Fault Message" pane, select an XML template to be used to pass back to clients when an error condition occurs.

12 To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each. Repeat as many times as desired. To remove an input XML template, select an entry and click **Delete**.

13 Select **Next**. The Connection Info panel of the Create a New J.D. Edwards Component wizard is displayed.


Create a New J.D. Edwards Component

Specify which Connection you wish to use for this Component or Service. To change any connection parameters, you must change them in the Connection Resource object or create a new Connection Resource of the same type with different parameters.

Connection	JDEdwardsConnection
Repository directory	ining\JDEdwards\JDERepository
User ID	JDE
Password	***
JDE Environment	DV7333
Application	XYZ
Server IP address	pvision01
Server Port	6009
Maximum Pool Size	10

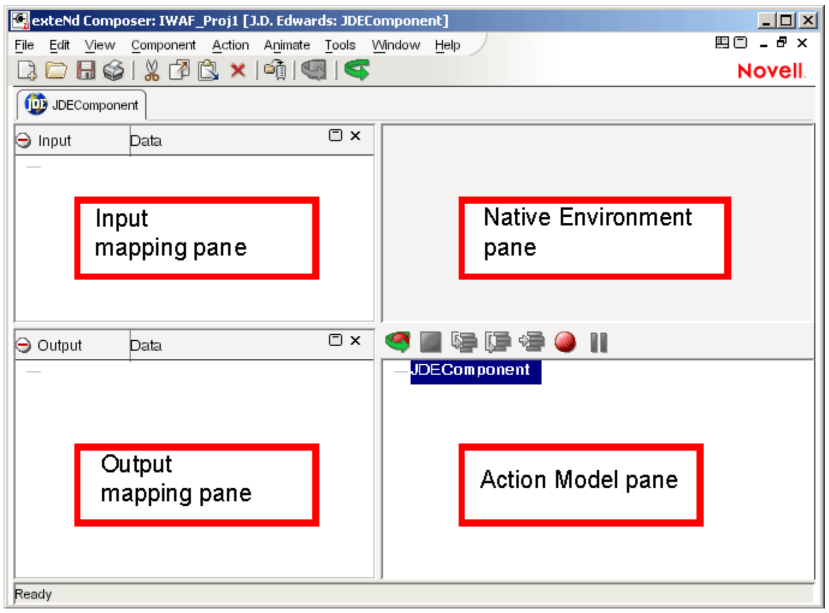
14 Select a J.D. Edwards connection from the **Connection** list. The Connection list displays the names of the J.D. Edwards connection resources that have been defined in this project.

15 Select **Finish**. The component is created and the J.D. Edwards Component Editor is displayed.

About the J.D. Edwards OneWorld Component Editor Window

The J.D. Edwards Component Editor includes all of the functionality of the XML Map Component Editor. It contains mapping panes for Input and Output XML documents, as well as an Action Model.

The J.D. Edwards Component Editor also includes a Native Environment pane, which appears as a grey pane until you create a J.D. Edwards Request action, at which time it will show a tabs for Request and Response panes with corresponding XML trees.



The Native Environment pane shown in the illustration is blank, because no J.D. Edwards Request action has been created.

Creating Actions in the Component Editor

You can create all the normal Composer actions in the Action Model of your J.D. Edwards component (e.g., XML Map, Function, Log, Send Mail). In addition, you can create a J.D. Edwards Request action.

The J.D. Edwards Request action communicates requests (XML request documents) to your J.D. Edwards system and fetches responses back from the same system.

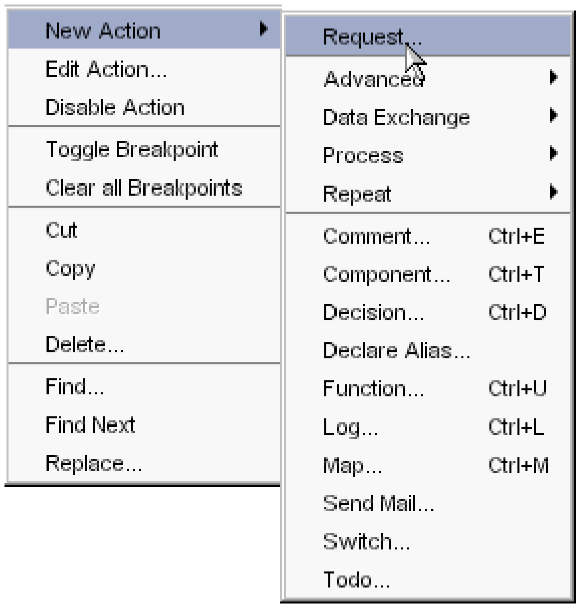
The Composer GUI for creating a J.D. Edwards Request action allows you to

- Specify the type of J.D. Edwards business function that you want to invoke
- Automatically generate XML schemas (request and response) for the particular business function that you want to use
- Use the generated schemas to fine-tune the structure of the actual request document that you want to use (a request document that conforms to the schema but reflects your own “setup” choices and initialization parameters, etc.)
- Automatically generate XML request and response documents that conform to the schemas and choices that you’ve made

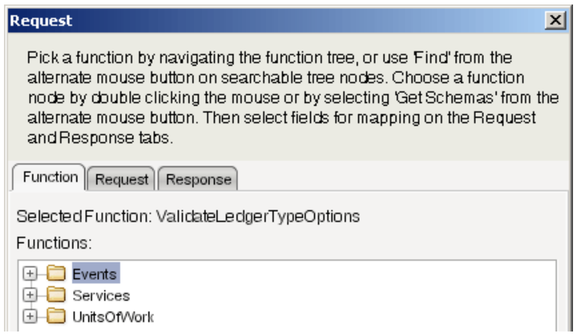
You will see how this works in the following procedure.

➢ To Create a J.D. Edwards Request Action

1 Right-click in the Action Model at the location in which you want the new action to appear, then select **New Action > Request**.



The Function pane of the Request dialog box is displayed. You use this pane to browse the available J.D. Edwards functions.


Request

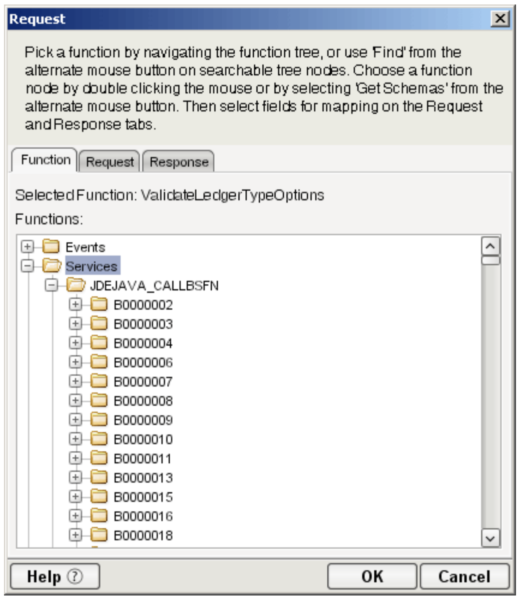
Pick a function by navigating the function tree, or use 'Find' from the alternate mouse button on searchable tree nodes. Choose a function node by double clicking the mouse or by selecting 'Get Schemas' from the alternate mouse button. Then select fields for mapping on the Request and Response tabs.

Selected Function: ValidateLedgerTypeOptions

Functions:

2 Click on the plus (+) sign to the left of a node in the tree view to expand the tree view for that node.

NOTE: If no nodes are displayed on the Function page, there may be an incorrect parameter in your connection resource. Check the parameters needed for connecting to your J.D. Edwards OneWorld server, then update your connection resource if necessary (see "Creating a Connection Resource" on page 19).


Request

Pick a function by navigating the function tree, or use 'Find' from the alternate mouse button on searchable tree nodes. Choose a function node by double clicking the mouse or by selecting 'Get Schemas' from the alternate mouse button. Then select fields for mapping on the Request and Response tabs.

Selected Function: ValidateLedgerTypeOptions

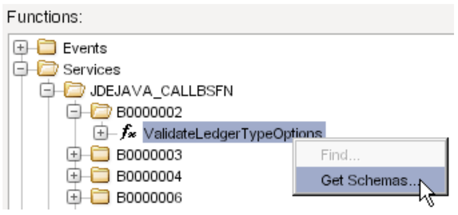
Functions:

Position the mouse pointer over a node in the tree view to get ToolTip information about the node. ToolTips display the following information, depending on the node type:

- Folder: “type - Folder”
- Searchable Folder nodes: “type - Service”
- Function nodes: “type - Operation”

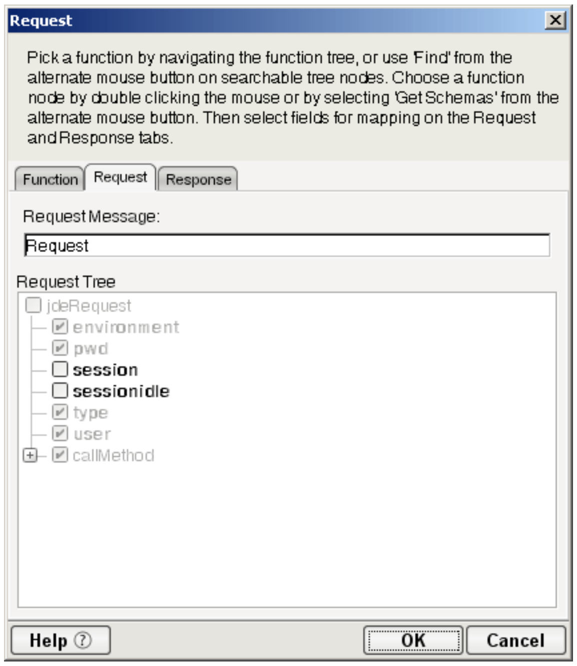
3 Click on the plus sign to the left of a node to see the contents of the node.

4 Right-click on the desired operation, then select **Get Schemas** from the context menu.



After a few seconds, the Request and Response tabs of the Request dialog box are enabled.

5 Select the **Request** tab. The Request pane is displayed.



Pick a function by navigating the function tree, or use 'Find' from the alternate mouse button on searchable tree nodes. Choose a function node by double clicking the mouse or by selecting 'Get Schemas' from the alternate mouse button. Then select fields for mapping on the Request and Response tabs.

Request Message:

Request

Request Tree

- jdeRequest
  - environment
  - pwd
  - session
  - sessionidle
  - type
  - user
  - callMethod

6 Type a name for the request message in the **Request Message** field, or accept the default name, as shown in the preceding illustration.

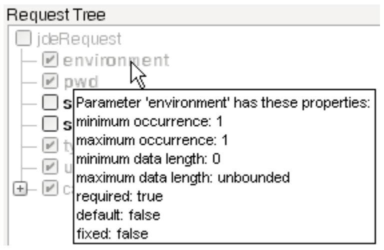
7 Expand the nodes of the Request Tree by clicking on the plus signs to the left of the nodes.

8 Select the elements that you want to include in the Request document by selecting or deselecting the check boxes in the node tree. Some nodes are greyed out (disabled). This is because the document schema determines the nodes that you can edit.

Composer strictly enforces schema rules. In other words, Composer will not let you specify an invalid request document. Items that you *should not edit* are disabled, and items that you can edit have context-menus that non-customizable nodes do not have. For example:

- *Mandatory* elements and attributes are marked *selected* and *disabled*.
- All child nodes are left unselected and disabled if the parent node is not selected.
- A parent element, once selected, will autoselect mandatory elements and attributes of its children.
- When a parent node is selected, you will need to select *optional* elements *manually.*

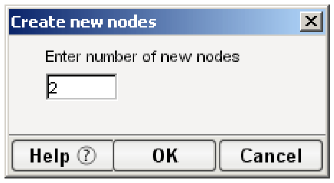
ToolTips are an important aid in using the Request and Response panes, because they display document-structure rules from the request and response document schema. Familiarize yourself with the ToolTip feature by letting the mouse hover over various nodes, as shown in the following illustration.



Parameter 'environment' has these properties:
minimum occurrence: 1
maximum occurrence: 1
minimum data length: 0
maximum data length: unbounded
required: true
default: false
fixed: false

NOTE: The ToolTips that appear in this tree are also available in the Native Environment pane.

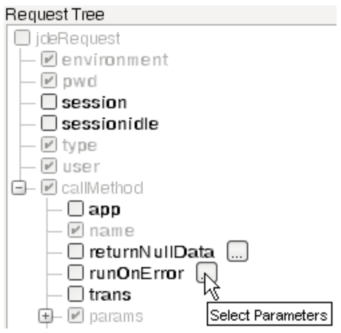
9 If a node has a **Maximum Occurrence** property of greater than one, a command called “Create new...” is available by right-clicking on the node. Selecting that command brings up the following dialog box:



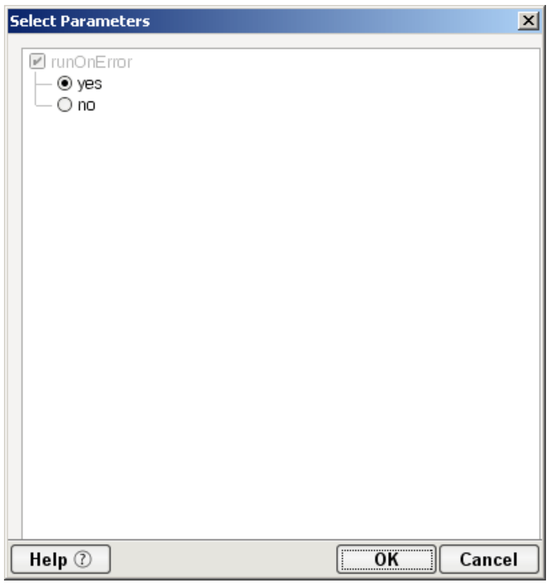
Enter the number of additional instances of this node-type that you want to insert in the document, then click **OK**. New nodes are added to the document structure.

CAUTION: There is no Undo for this operation.

10 If an enumeration button (...) is displayed to the right of a given node, you can click the button to choose from the list of appropriate (schema-allowed) values for that enumeration.

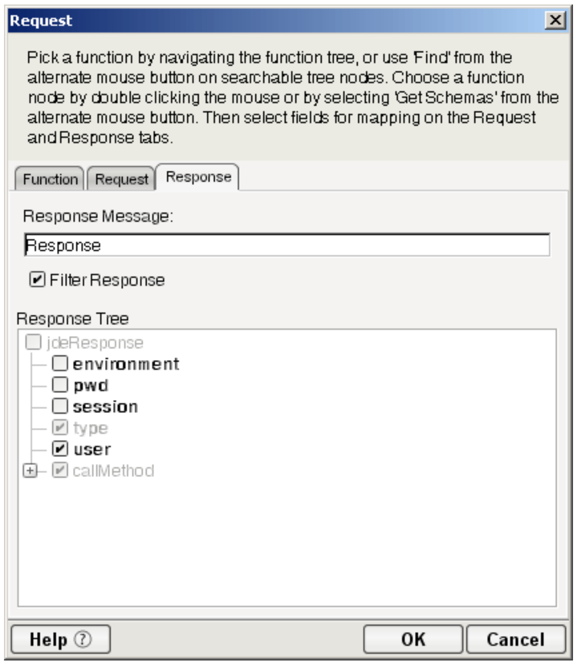


You can customize the operation of the selected method by selecting a parameter from an enumeration, as shown in the following illustration.



11 Visit all of the nodes in the request tree that are of interest to you, and use the Composer user interface features described in the preceding steps to customize the request document structure.

12 Click the **Response** tab at the top of the dialog box to bring the Response document pane forward. This pane shows a tree view of the Response document, similar to the one shown for the Request document. The same user interface features that applied to the Request document pane apply to the Response document pane.


Request

Pick a function by navigating the function tree, or use 'Find' from the alternate mouse button on searchable tree nodes. Choose a function node by double clicking the mouse or by selecting 'Get Schemas' from the alternate mouse button. Then select fields for mapping on the Request and Response tabs.

Function | Request | Response

Response Message: Response

[✓] Filter Response

Response Tree
- [ ] jdeResponse
  - [ ] environment
  - [ ] pwd
  - [ ] session
  - [✓] type
  - [✓] user
  - [✓] callMethod

13 Type a name for the response message in the **Response Message** field, or accept the default response message name.

14 Select or deselect **Filter Response**, as desired. **Filter Response** is selected by default. If deselected, the tree will be disabled and greyed (no longer editable), indicating that a default schema configuration will be used. If selected, you can customize the structure of the request document by selecting the nodes to be included in the response document.

15 Visit all of the nodes in the Response document tree and customize the settings as desired.

16 Select **OK**. The dialog closes and a new action appears in the Action Model of your component. The Native Environment Pane changes to show the Request and Response tabs along with tree views of the request and response documents.

17 To map a node of the Input document to a node in the Request document, drag a node from the Input pane and drop it on a node in the Request tab of the Native Environment pane. This automatically creates a new XML Map Action in the Action Model.

18 To map a node of the Response document to a node in the Output document, drag a node from the Response tab of the Native Environment pane and drop it on a node in the Output pane. This automatically creates a new XML Map Action in the Action Model. You can cut, copy, or delete the actions once they appear in the Action Model pane.

19 Add or remove actions from the Action Model to create the desired business logic.

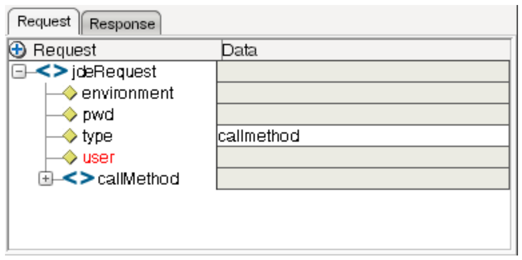
Returning to Schema-Edit Mode

Once you have created a Request Action in your Action Model (see "Creating Actions in the Component Editor" on page 27), you can go back and edit the Request and Response schema trees by double-clicking the Request Action in the Action Model. Double-clicking causes the Request dialog box to be displayed. You can use the Request and Response tabs to navigate the request and response document schemas and change your customizations as desired. When you exit from of the dialog box, changes are shown in the Native Environment Pane.

NOTE: If you manually edit the Request or Response documents in the Native Environment Pane (see "Manually Editing the Request and Response Documents" on page 33), you will not be able to reopen the Request Action dialog box. Avoid manually editing request/response documents. Instead, make modifications through careful use of the XML Map Action or by using Request.createXPath( ) and DOM methods in a Function Action.

Request and Response Documents

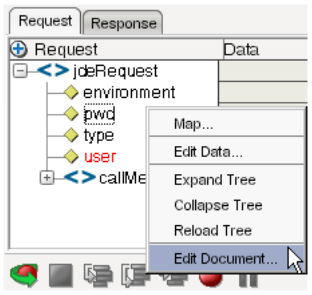
After you have created a Request Action (see "Creating Actions in the Component Editor" on page 27), the Native Environment Pane portion of the Composer editor view displays Request and Response documents in tree-view form, as shown in the following illustration.



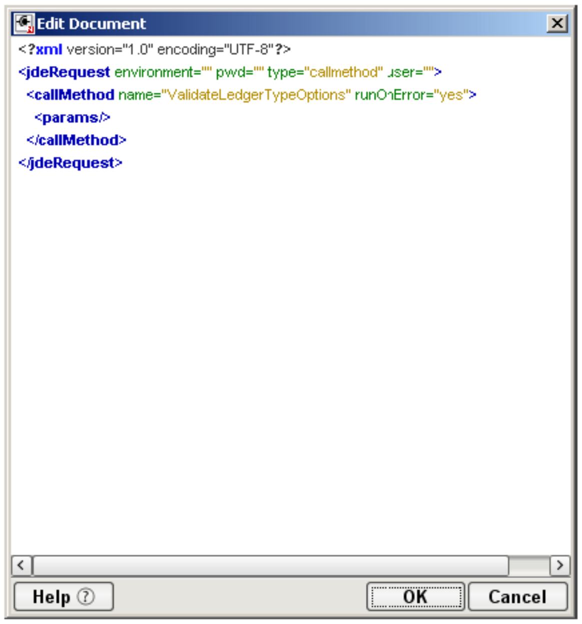
You can drag and drop data from the Input document to the Request document to create XML Map actions. You also can use drag-and-drop mapping to map from Response to Output. In the preceding illustration, a node is displayed in red. The red color indicates that data has been mapped to the node.

Manually Editing the Request and Response Documents

If you right-click anywhere inside the Request or Response tree views in the Native Environment Pane, a menu is displayed.



The **Edit Document** command in this menu opens the document in a text-editor, as shown in the following illustration.

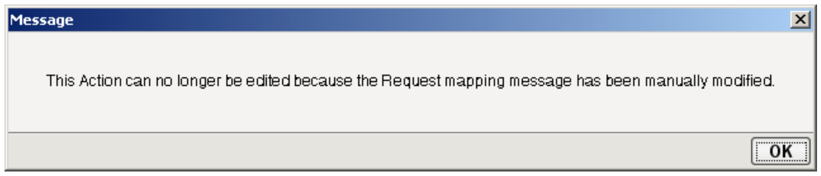


```
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest environment="" pwd="" type="callmethod" user="">
  <callMethod name="ValidateLedgerTypeOptions" runOnError="yes">
    <params/>
  </callMethod>
</jdeRequest>
```

In this window, you can manually edit the Request or Response document.

CAUTION: *If you manually edit the document, it may no longer conform to the original schema. As a result, you will no longer be able to open the document in the Request Action dialog box.*

If you have manually edited a Request or Response document, double-clicking on the corresponding Request Action in the Action Model will cause the following message to be displayed:



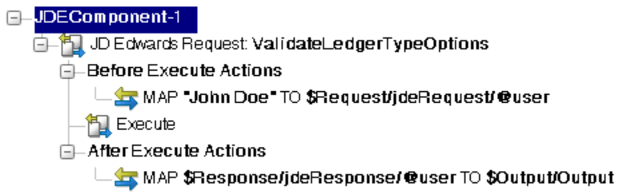
This Action can no longer be edited because the Request mapping message has been manually modified.

If this message appears, and you want to work with the document using the Request Action dialog box, you must recreate the action in order to see the original Request or Response document. Manual edits will be lost.

To prevent this situation from occurring, *avoid making manual edits to Request or Response documents*. Instead use XML Map Actions and Function actions to modify documents or node contents. Be aware that doing so may violate schema constraints. This will not necessarily cause runtime problems for your component when executing in the Composer Enterprise Server environment, because Composer does not validate the request document against the schema at runtime. However, your J.D. Edwards OneWorld system may validate incoming requests, in which case a manually-edited Request document can cause errors that are difficult to troubleshoot.

“Before Execute” and “After Execute” Actions

Whenever a J.D. Edwards Request action is created in the Action Model, Composer adds two additional lines to the action list: Before Execute Actions and After Execute Actions. These are header labels (grouping labels) for blocks of actions that occur *before* the request is sent to the J.D. Edwards OneWorld system, or *after* a response document has come back, respectively. Typically you will want to map data fields from Input to Request and have those actions be grouped under the “Before Execute Actions” header. Likewise, any map actions or other action logic that you need to perform on the Response document after the J.D. Edwards OneWorld system executes your request should be grouped under the “After Execute Actions” heading.



```
JDEComponent-1
  JD Edwards Request: ValidateLedgerTypeOptions
    Before Execute Actions
      MAP "John Doe" TO $Request/jdeRequest/@user
    Execute
    After Execute Actions
      MAP $Response/jdeResponse/@user TO $Output/Output
```

Creating J.D. Edwards OneWorld Services

This section describes how to use the exteNd Composer™ Connect for J.D. Edwards OneWorld to create a J.D. Edwards OneWorld Service component to connect to J.D. Edwards OneWorld and listen for events. As part of the process of creating a J.D. Edwards OneWorld Service, you can select an existing J.D. Edwards OneWorld connection resource, or you can create a new one. If you create the connection beforehand, it is available for use by any J.D. Edwards OneWorld Services in the current project. *If you have not already created at least one J.D. Edwards OneWorld connection resource in the current project, you will be prompted to do so when you try to create a J.D. Edwards OneWorld Service.*

NOTE: You can still create a J.D. Edwards OneWorld Service without first creating a connection resource, but you won’t be able to use any debug features that depend on a live connection.

➢ **To Create a New J.D. Edwards OneWorld Service:**

1 Select **File** > **New** > **xObject**. The New xObject dialog box is displayed.

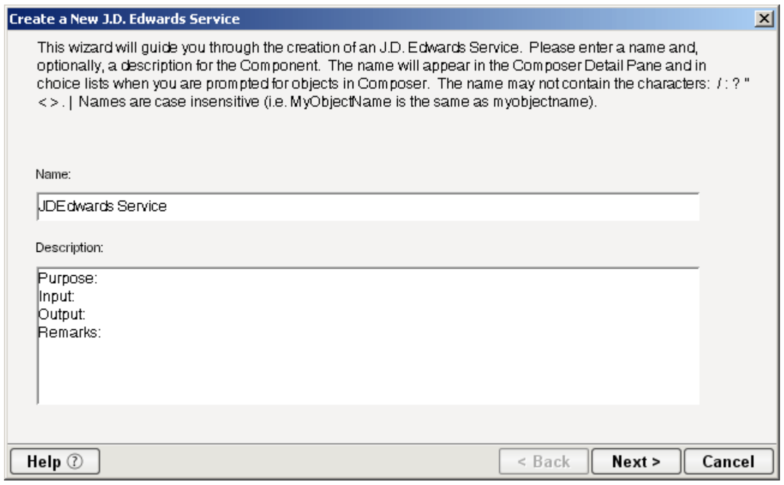
NOTE: Alternatively, under **Service** in the Composer Navigator pane (Explorer view), you can highlight **J.D. Edwards OneWorld Service**, click the right mouse button, then select **New**.

2 Select the **Process/Service** tab.

3 Double-click on “J.D. Edwards OneWorld Service.”

If you have not previously defined a connection resource, you are prompted to do so now (see “Creating a Connection Resource” on page 19).

If you have already defined a connection resource, the “Create a New J.D. Edwards OneWorld Service” Wizard is displayed.



This wizard will guide you through the creation of an J.D. Edwards Service. Please enter a name and, optionally, a description for the Component. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: / : ? " < > . | Names are case insensitive (i.e. MyObjectName is the same as myobjectname).

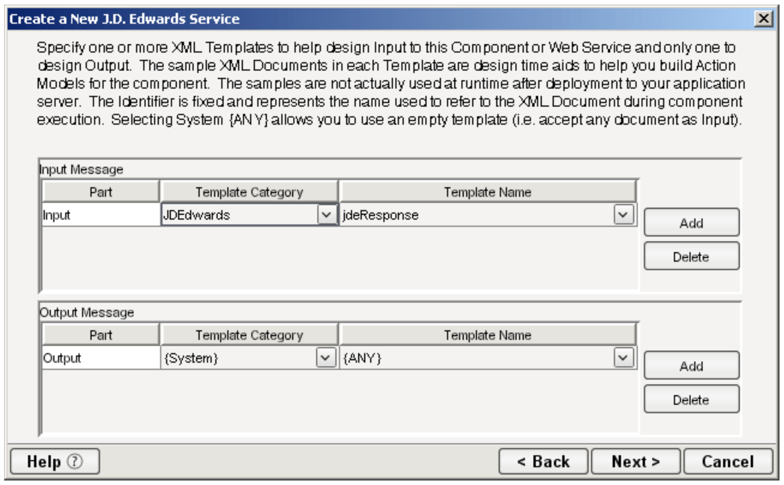
Name: JDEdwards Service

Description:
Purpose:
Input:
Output:
Remarks:

4 Type a **Name** for the new J.D. Edwards OneWorld Service.

5 Optionally, type **Description** text.

6 Select **Next**. The XML Input/Output Message Property Info panel of the New J.D. Edwards OneWorld Service Wizard is displayed.



Specify one or more XML Templates to help design Input to this Component or Web Service and only one to design Output. The sample XML Documents in each Template are design time aids to help you build Action Models for the component. The samples are not actually used at runtime after deployment to your application server. The Identifier is fixed and represents the name used to refer to the XML Document during component execution. Selecting System {ANY} allows you to use an empty template (i.e. accept any document as Input).

Input Message

Part	Template Category	Template Name
Input	JDEdwards	jdeResponse

Output Message

Part	Template Category	Template Name
Output	{System}	{ANY}

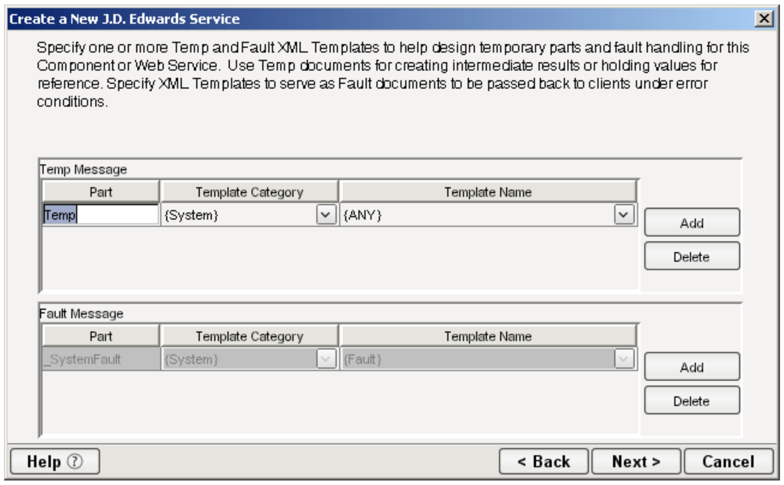
7 Specify the Input and Output templates as follows:

- Type a name for the template under **Part** if you wish the name to appear in the DOM as something other than “Input”
- Select a **Template Category** if it is different than the default category.
- Select a **Template Name** from the list of XML templates in the selected **Template Category**.
- To add additional input XML templates, click **Add**, then choose a **Template Category** and **Template Name** for each.
- To remove an input XML template, select an entry and click **Delete**.

8 Select an XML template for use as an Output DOM using the procedure described in the previous step.

NOTE: You can specify an input or output XML template that contains no structure by selecting {System}{ANY} as the Input or Output template. For more information, see “Creating an Output Document without Using a Template” in the *Novell exteNd Composer User’s Guide*.

9 Select **Next**. The Temp and Fault XML Template panel is displayed.


Create a New J.D. Edwards Service

Specify one or more Temp and Fault XML Templates to help design temporary parts and fault handling for this Component or Web Service. Use Temp documents for creating intermediate results or holding values for reference. Specify XML Templates to serve as Fault documents to be passed back to clients under error conditions.

Temp Message

Part	Template Category	Template Name
Temp	{System}	{ANY}

Fault Message

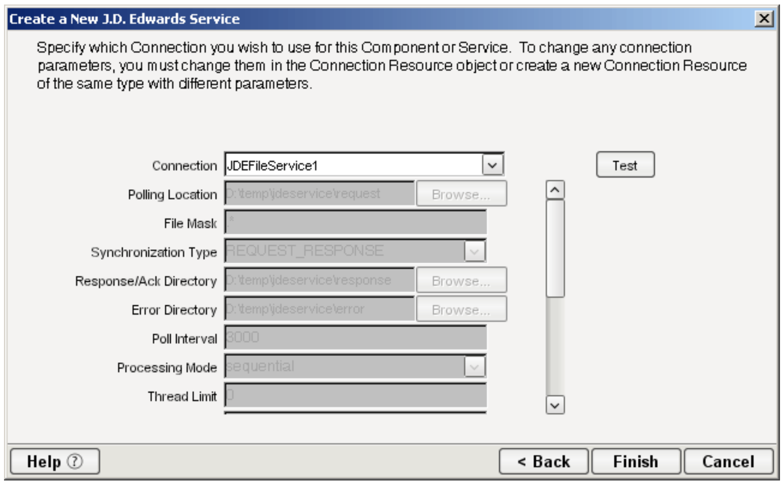
Part	Template Category	Template Name
_SystemFault	{System}	{Fault}

10 If desired, specify a template to be used as a scratch pad under the “Temp Message” pane of the dialog box. This can be useful if you need a place to hold values that will be used temporarily during the execution of your component. Select a **Template Category** if it is different than the default category. Then select a **Template Name** from the list of XML templates in the selected Template Category.

11 Under the “Fault Message” pane, select an XML template to be used to pass back to clients when an error condition occurs.

12 To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each. Repeat as many times as desired. To remove an input XML template, select an entry and click **Delete**.

13 Select **Next**. The Connection Info panel of the Create a New J.D. Edwards OneWorld Service wizard is displayed.


Create a New J.D. Edwards Service

Specify which Connection you wish to use for this Component or Service. To change any connection parameters, you must change them in the Connection Resource object or create a new Connection Resource of the same type with different parameters.

Connection	JDEFileService1
Polling Location	D:\temp\jdeservice\request
File Mask	*
Synchronization Type	REQUEST_RESPONSE
Response/Ack Directory	D:\temp\jdeservice\response
Error Directory	D:\temp\jdeservice\error
Poll Interval	3000
Processing Mode	sequential
Thread Limit	0

14 Select a J.D. Edwards OneWorld File or TCP Service connection from the **Connection** list. The Connection list displays the names of the J.D. Edwards OneWorld connection resources that have been defined in this project.

15 Select **Finish**. The component is created and the J.D. Edwards OneWorld Service Editor is displayed. The features of this window are identical to the features described in “About the J.D. Edwards OneWorld Component Editor Window” on page 27. The following sections describe the steps required to create a J.D. Edwards OneWorld Service Action.

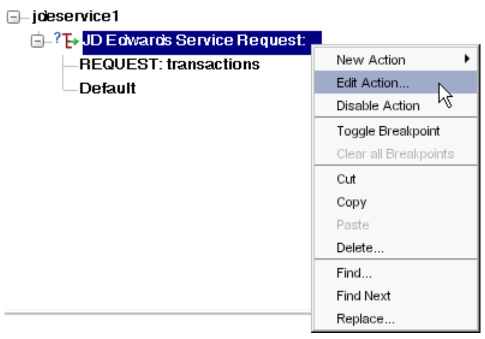
Creating J.D. Edwards OneWorld Service Actions

You can create all the normal Composer actions in the Action Model of your J.D. Edwards OneWorld Service (e.g., XML Map, Function, Log, Send Mail). In addition, you can create a J.D. Edwards OneWorld Service Request action.

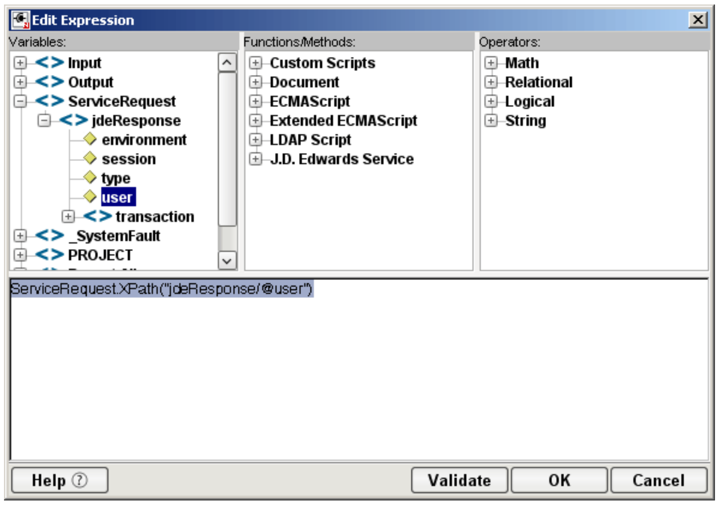
The J.D. Edwards OneWorld Service Request action is similar to a Switch action (see "The Switch Action" in the in the *Novell exteNd Composer User's Guide).* A Switch action allows program control to branch to a block of actions based on a match between an input value and a Case value. In a J.D. Edwards OneWorld Service Request, Composer receives a DOM, and compares a series of cases against the DOM. If an exact match occurs between the DOM and a case, execution branches to the actions listed underneath the case in the Action model. Cases are tested in the order listed; and once a match is found, execution of the match logic precludes execution of any other logic in the J.D. Edwards OneWorld Service Request.

➢ To Create a J.D. Edwards OneWorld Service Request Action

1. In the Action Model, right-click on J.D. Edwards OneWorld Service Request:



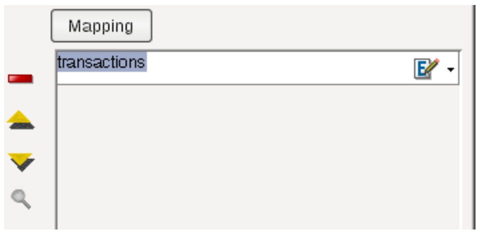
2. Select **Edit Action**. The Service Request dialog box is displayed.
3. In the Function tree, click on the plus (+) sign to the left of the Messages node to expand the tree view for that node.
4. Right-click on the message name that you want to use for input and select **Add Case,** or double-click on the message name. The Adapter Service Request dialog box is displayed. This dialog box is similar to the J.D. Edwards OneWorld Request dialog box (see "Creating Actions in the Component Editor" on page 27).
5. Select or deselect **Filter Request**, as desired. Filter Request is selected by default. If deselected, the tree will be disabled and greyed (no longer editable), indicating that a default schema configuration will be used. If selected, you can customize the structure of the request document by selecting the nodes to be included in the request document.
6. Expand the nodes of the Service Request Tree by clicking on the plus signs to the left of the nodes.
7. Select the elements that you want to include in the Service Request document by selecting or deselecting the check boxes in the node tree. Some nodes are greyed out (disabled). This is because the document schema determines the nodes that you can edit.
8. If the **Service Response** tab is enabled (this depends on the Synchronization Type setting in the Connection resource), click the **Service Response** tab at the top of the dialog box to bring the Service Response document pane forward. This pane shows a tree view of the Response document, similar to the one shown for the Request document. The same user interface features that applied to the Request document pane apply to the Response document pane.
9. Visit all of the nodes in the Response document tree and customize the settings as desired.
10. Click the **Case Expression** tab at the top of the dialog box to bring the Case Expression pane forward. This pane provides an Expression Builder that you use to build a case expression.



```
ServiceRequest.XPath("jdeResponse/@user")
```

11 Type the static string values or the ECMAScript expressions that will be checked against the input DOM.

12 Select the OK button. The Service Request dialog box is displayed.



13 To add another case, repeat Step 3 through Step 12. The cases are listed from top to bottom in the order in which they are evaluated. Each Case value will be checked in turn, in the order you list them.

- For optimal performance, list the most likely matches first.
- You can change the order in which the cases are listed by clicking on the name of a case, then clicking the up or down triangle icons.
- To edit a case, select the name of the case, then press the **Mapping** button. You can also select the name of the case, then select the Expression Builder button.

14 When you have finished adding cases, select **OK.** The Service Request dialog box closes and a new action appears in the Action Model of your component. The Native Environment Pane changes to show the Request and Response (if the Synchronization Type is set to REQUEST_RESPONSE in the Connection resource) tabs along with tree views of the request and response documents.

15 In the Action Model, add actions to be performed to each case to create the desired business logic.

- To map a node of the Input document to a node in the Request document, drag a node from the Input pane and drop it on a node in the Request tab of the Native Environment pane. This automatically creates a new XML Map Action in the Action Model.
- To map a node of the Response document to a node in the Output document, drag a node from the Response tab of the Native Environment pane and drop it on a node in the Output pane. This automatically creates a new XML Map Action in the Action Model. You can cut, copy, or delete the actions once they appear in the Action Model pane.
- When all of the Actions for a case have been evaluated, Composer sets the Service Response DOM.

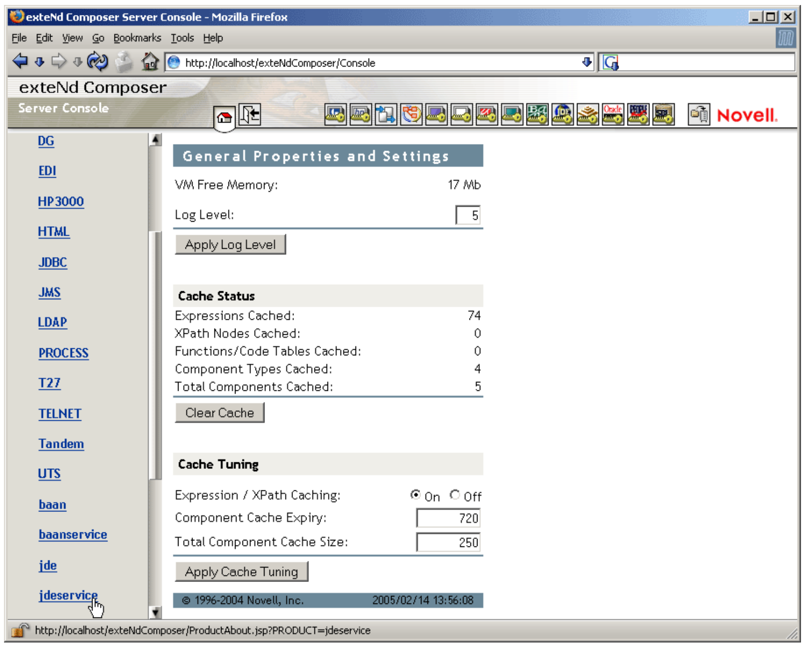
- The final case in the Action model is always labeled Default. This case is generated automatically and cannot be removed. Actions placed under Default are executed if and only if the none of the other cases are matched. While you are not required to place actions under Default, it is good programming practice to have at least some kind of fallback logic for the Default case, even if it's only a Log action or a Raise Error action.
- The Request and Response document displays for Service Requests work similarly to the displays for Requests (see "Request and Response Documents" on page 33).

Managing Deployed J.D. Edwards OneWorld Services

Once a project containing J.D. Edwards OneWorld Services has been deployed (see "Deploying Your Project" in the *Novell exteNd Composer User's Guide)*, the Listener objects actively listen for messages each time you start your server. To manually start and stop these services you need to use the exteNd J.D. Edwards OneWorld Services Console. This browser-based console allows you to see the list of J.D. Edwards OneWorld Services, the status of each service (running or not running), the running tally (Count) of messages received, other administrative information, and a Start/Stop button.

➢ To display the exteNd J.D. Edwards OneWorld Services Console:

1. Make sure that your application server is running.
2. From the Windows Start menu, select **Programs > Novell Extend 5.2 > Composer > Composer Server Console**. A dialog box for entering the application server administrator ID and password is displayed.
3. Type your administrator ID and password, then select **OK**. The Composer Server Console page is displayed.
4. Scroll down the **About Products** list on the left side of the page until you see the link titled "jdeservice".



http://localhost/exteNdComposer/Console

General Properties and Settings

VM Free Memory: 17 Mb
Log Level: 5

Cache Status
Expressions Cached: 74
XPath Nodes Cached: 0
Functions/Code Tables Cached: 0
Component Types Cached: 4
Total Components Cached: 5

Cache Tuning
Expression / XPath Caching: [✓ On] Off
Component Cache Expiry: 720
Total Component Cache Size: 250

© 1996-2004 Novell, Inc. 2005/02/14 13:56:08

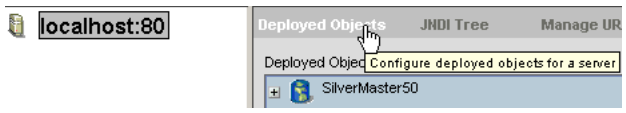
http://localhost/exteNdComposer/ProductAbout.jsp?PRODUCT=jdeservice

5. Select the "jdeservice" link. A page is displayed that provides information (e.g., version, license number, copyright) about the Connect. It also displays a Console button.
6. Select the Console button. The J.D. Edwards OneWorld Service console page is displayed. This page lists any J.D. Edwards OneWorld Services that have been deployed.
7. To stop a J.D. Edwards OneWorld Service, select the appropriate **Stop** button (the button will then change to Start).

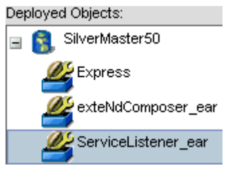
NOTE: If messages are being handled by a service at the time of the **Stop** command, there may be some delay before the service actually exits. Select the **Refresh** button periodically until the "Running" column of the console says No for the service.

➢ **To undeploy a J.D. Edwards OneWorld Service:**

1. Make sure that your application server is running.
2. From the Windows Start menu, select **Programs > Novell Extend 5.2 > AppServer > Server Management Console**.
3. Login as an administrator (**File > Login**).
4. Select the **Deployment** icon ( )from the toolbar.
5. Select **Deployed Objects**:



6. Expand the database containing the deployed objects that you want to manage:



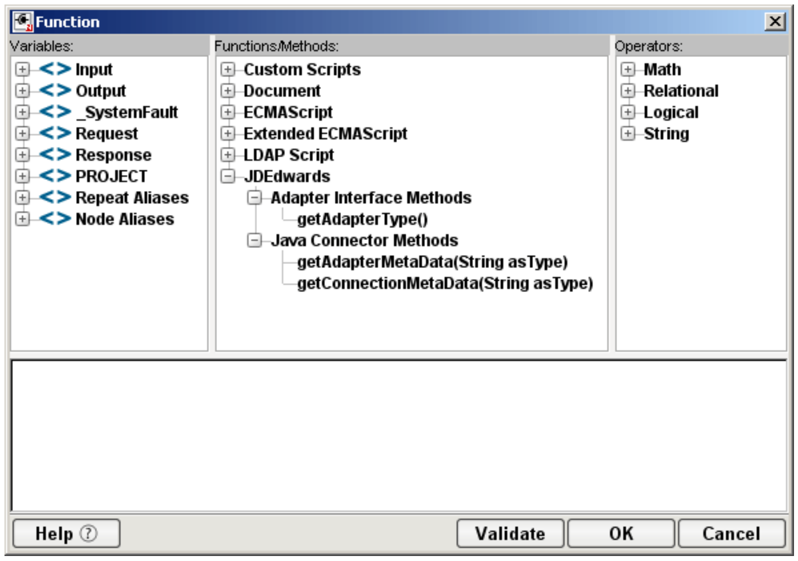
7. Select the object that you want to undeploy.
8. Select the **Undeploy** button.

ECMAScript Extensions

See Appendix A to this guide for a list of ECMAScript extension methods that can be used with the J.D. Edwards OneWorld Connect.

A ECMAScript Methods

The J2EE Connector framework of the exteNd Composer Connect for J.D. Edwards OneWorld implements a number of ECMAScript extension methods that are available for use through the Expression Builder (which is available in Function Actions and elsewhere). You will find the methods listed in the Functions/Methods pane of the Expression Builder:



ToolTip help is available for each method.

For more information about working with the Expression Builder, see “Custom Scripting and XPath Logic in exteNd Composer” in the *Novell exteNd Composer User’s Guide*.

Adapter Interface Methods

getAdapterType

This method calls the IGNVAdapterInterface.getAdapterType(), and returns the adapter type name (e.g., “jde”).

Syntax:
getAdapterType()

Connection Interface Methods

The following CCI API metadata-wrapped methods are published through the Expression Builder.

getAdapterMetaData

This method wraps calls on the javax.resource.cci.ResourceAdapterMetaData class.

Syntax:
getAdapterMetaData(*parameter*)

Parameter	Description
NAME	Returns the name of the adapter (e.g., “iWay JCA adapter”).
EXECUTEWITHI	Returns true if the implementation class for the Interaction interface implements public Record execute method; otherwise the method returns false.
EXECUTEWITHIO	Returns true if the implementation class for the Interaction interface implements public boolean execute method; otherwise the method returns false.
VENDOR	Returns the name of the vendor that has provided the resource adapter (e.g., “iWay Software, Inc.”).
VERSION	Returns the version of the resource adapter (e.g., “5.5”).
INTERACTIONSPECS	Returns the fully-qualified name of the InteractionSpec type supported by this resource adapter (e.g., “com.ibi.afjca.IWAFInteractionSpec”).
SPECVERSION	Returns a string representation of the version of the connector architecture specification that is supported by the resource adapter (e.g., “1.0”).
LOCALTX	Returns true if the resource adapter implements the LocalTransaction interface and supports local transaction demarcation on the underlying EIS (Enterprise Information System) instance through the LocalTransaction interface.

getConnectionMetaData

This method wraps calls on the javax.resource.cci.ConnectionMetaData class.

Syntax:
getConnectionMetaData(*parameter*)

Parameter	Description
EISPRODUCTNAME	Returns the product name of the underlying EIS instance (e.g., “JDEdwards”).
EISPRODUCTVERSION	Returns product version of the underlying EIS instance (e.g., “OneWorld”).
USER	Returns the user name for an active connection as known to the underlying EIS instance.

Additional Methods

The following ECMA-wrapped methods are not displayed by the Expression Builder user interface, but you can use them in the Expression Builder by typing them directly into expressions.

getWarnings()

Returns line separated list of javax.resource.cci.ResourceWarning for the javax.resource.cci.Interaction.

clearWarnings()

Calls javax.resource.cci.Interaction.clearWarnings().

getLastError()

Returns last error, such as a javax.resource.ResourceException.

Index